PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

A NOVEL TECHNIQUE FOR FPGA IP PROTECTION

Sunil Malipatlolla and Sorin A. Huss

Center for Advanced Security Research Darmstadt (CASED)
Technische Universität Darmstadt, Germany

ABSTRACT

The configuration data sequence of a field programmable gate array (FPGA) is an intellectual property (IP) of the original designer. With the increase in deployment of FPGAs in modern embedded systems, the IP protection of FPGA hardware designs has become a necessary requirement for many IP vendors. There have been already many proposals to overcome this problem using symmetric encryption techniques but these methods need a cryptographic key to be stored in a non-volatile memory located on FPGA or in a battery-backed RAM (Random Access Memory) as done in some of the current FPGAs. The expenses with the proposed methods are, occupation of larger area on FPGA in the former case and limited lifetime of the device in the latter. In contrast, we propose a novel method which combines the dynamic partial reconfiguration (dynamic PR) feature of an SRAM-based FPGA with the public key cryptography (PKC) to protect the FPGA configuration files without the need to store any keys on FPGA. Using our method, not only the high-end FPGAs but also the low-end FPGAs with partial reconfiguration capabilities are secured. The proposed method has been implemented on a Xilinx Virtex-5 FPGA platform.

Index Terms— FPGA, bitstream, Public Key Cryptography, Dynamic Partial Reconfiguration, IP, Embedded Systems

1. INTRODUCTION

Nowadays, static-random-access-memory-based (SRAM-based) FPGAs are becoming increasingly popular as building blocks of electronic systems because of advantages such as easy design modification (reconfigurability), rapid prototyping, economical cost for low volume production, lower startup cost in comparison to fully-customized application-specific-integrated-circuits (ASICs), and availability of sophisticated design and debugging tools. Applications of FPGAs in the area of consumer electronics include, for example, television circuits, communication & video processing devices and software-defined radios.

Since FPGAs are becoming so important for the electronic industry, it is necessary to think about the security of FPGA-based systems. Two possible security measures include are, protecting the FPGA data and the FPGA design itself. In the former case it is necessary to protect the FPGA application i.e., the data inside the circuit and the data transferred to/from the peripheral circuits during the communication. Whereas in the latter, the concerns are against cloning and reverse engineering which is the IP protection problem. Concerning SRAM-based FPGAs it corresponds to the way to protect the bitstream so the FPGA configuration. In essence, the problem of design security is simple, the designer doesn't want that a competitor could be able to pirate his design.

There are two types of piracy: cloning and reverse engineering. Cloning is when a competitor makes a copy of the design, and when he is able to make a copy of the pirated system. With FPGAs it is very simple to clone an unprotected design as the bitstream can be copied to another FPGA's configuration memory. In case of reverse engineering the design is copied by reconstructing a schematic or netlist level representation; in this process he understands how the design works and how to improve it, or to modify it with malicious intent. So the reverse engineering is more serious than cloning. These two correspond to different attacks, and the design security must protect the system against both these attacks. There are two types of attacks: non-invasive and invasive.

- **The non-invasive attacks** gather all the methods that use external means. For example the attackers can use all the possibilities of the circuit inputs in order to obtain all the different outputs and draw the system truth table, this method is called "black box" attack. In case of an SRAM-based FPGA a simple attack method can be, intercepting the bitstream between the root ROM and the FPGA when the power is switched on. More complex attacks can use power and electromagnetic changes and measures like the simple or differential power analysis [1].
- **The invasive attacks** (- or **physical attacks**) are characterized by the necessity to destroy the integrated circuit (component package) to study the

Thanks to CASED for funding.

chip (design inside the component) using some complex methods. For example, it is possible to use a laser cutter microscope in order to split the chip in several slices and understand the chip layout. These attacks can use sophisticated tools like optical microscope, mechanical probes and even focused ion beam (FIB). As these attacks use the weakness of the silicon technology, when they are possible, it is very hard to secure the system against them.

The papers [2] and [3] give some information about these different attacks. In this paper we consider the protection of FPGA configuration files against the non-invasive attacks only. The rest of this paper is organized as follows. Section 2 gives an overview about the related work done to address the problem of FPGA configuration file protection. Section 3 briefly explains the dynamic PR supported by the state of the art FPGAs and an overview about the PKC while Section 4 lists the objectives to be achieved with the assumed scenario and describes our own methodology to protect the FPGA bitstreams using dynamic PR and PKC. In section 5, an analysis of the implementation results is given while section 6 concludes the paper by giving an outlook into the future work.

2. RELATED WORK

There are generally two approaches possible to address the problem of FPGA IP protection. The first solution to protect the device against the piracy is the legal solution. The definition of efficient laws, the regulation and the management of intellectual properties are parts of this solution. The second proposal to improve the security level of SRAM-based FPGAs is by bitstream encryption. In this section we mostly address the related work done using the second approach only. For example, Xilinx Virtex series devices support configuration with an encrypted bitstream. Virtex devices have a built-in bitstream decryption unit on them. Virtex-II and Virtex-II Pro support Triple data encryption standard (Triple-DES) [4] with a 56-bit key, while Virtex-4 and Virtex-5 support AES [5] with a 256-bit key. The secret key is stored in a dedicated volatile memory inside the FPGA which must always be supplied with power through an external battery, which limits the lifetime of the device. Additionally, the on-board decryption unit and the corresponding key occupy a considerable amount of space which is very crucial in embedded system design.

In order to overcome the problem of an additional battery in Xilinx's solution, Tom Kean of the Algotronix society proposed ideas to store the cryptographic secret key on FPGA, like using laser to program a set of links during manufacture [6]. However, in his method the encryption and decryption circuits are embedded inside the FPGA which causes less available silicon area for developed applications. Also, the encryption and decryption circuits are fixed, so it is not possible to upgrade them. In contrast, Kun-Wah Yip et al. proposed the IP protection scheme using partial-encryption (PE) technique [7]. Their method argues that the PE technique outperforms the full-encryption technique in terms of the reverse engineering cost. Whereas, Jorge Guajardo et al. proposed a different scheme, using FPGA's intrinsic physical unclonable functions (PUF) and PKC for IP protection. Though their method uses PKC-based authentication protocol which does not need the private key to be stored on the FPGA, they did not make use of the advantages provided by partial reconfiguration. In addition the PUF implementation and its analysis on an FPGA is in itself a challenging task [8].

There are other techniques proposed for FPGA IP protection like watermarking as in John Lach et al., where they apply a watermark to the physical layout of a digital circuit when it is mapped onto an FPGA which uniquely identifies the circuit origin and yet difficult to detect [9]. In contrast, Tim Güneysu et al. used both public-key and symmetric key cryptography to dynamically protect the IP of circuits in configuration files. In their method the symmetric cryptography is hardwired and the public-key functionality is moved into a temporary configuration bitstream for a one-time setup procedure [10]. Also, Bossuet et al. proposed a scheme where an embedded key is accessible to the user logic and uses partial reconfiguration to encrypt and decrypt the bitstream [11]. An on-chip key is used to encrypt the main design's bitstream before storing it in a PROM where a decryption bitstream is also stored to decrypt the encrypted bitstream in the field. The flaw of this scheme is that if the key is accessible to the user logic anyone can read it and decrypt the bitstream.

As mentioned above, all the methods need a secret key to be stored on an FPGA which in itself is a challenge in SRAM-based FPGAs as the memory on these devices is volatile. In contrast, we can store the keys in a non-volatile memory placed on an FPGA, but this has two drawbacks: One being necessity for an extra space on FPGA, which is crucial for embedded systems as they are area constrained, and the other being the possible extraction of stored cryptographic keys by an attacker which makes the device less secure. However, to the best of our knowledge, the idea of using dynamic PR and PKC for FPGA bitstream protection has not yet been addressed. Our method utilizes the special feature of SRAM-based FPGAs, the partial reconfiguration, and the public key cryptography to protect the FPGA bitstreams.

The novelty of our method is that it does not need any fixed key storage for encryption and decryption of bitstreams as the keys are generated on the fly. There is no threat of the private key being stolen, as it is stored (temporarily) deep inside memory blocks which are erased when the device is turned off. Also, as the keys for encryption and decryption of bitstreams are generated on the FPGA, unlike the single symmetric key in previously referenced papers, and sent to the host for bitstream (IP) encryption it is possible to address the problem of loading IPs from different vendors. Different vendors can use on the fly generated keys to encrypt their IPs, before sending them to the FPGA for secure deployment in the field, which means that they will all be placed on a single System-on-Chip (SoC).

3. DYNAMIC PR IN SRAM-BASED FPGAS

3.1. Partial Reconfiguration Overview

Some of the SRAM-based FPGAs support a special feature called partial reconfiguration (PR) in which a portion of the FPGA's fabric is reconfigured while the rest resumes its work. The portion being reconfigured is the reconfigurable (dynamic) part and the portion resuming the work is the static part. If the configuration of the FPGA is changed at run-time i.e., the system is neither stopped nor switched off, then its called as dynamic PR. Additionally, if the system triggers the reconfiguration by itself then it is a self-reconfigurable system which does not require the use of internal FPGA infrastructure. The area of the FPGA that is reconfigured is called the partially reconfigurable region (PRR). A PRR typically consists of a number of configurable logic blocks (CLBs) and functional blocks. The module to be placed inside the PRR is called a partially reconfigurable module (PRM), which is the specific configuration of the PRR and at least two PRMs are needed per PRR. In many cases the assignment of PRMs to PRR is fixed (non-relocatable) though in principle, a PRM may be configured to different PRRs.

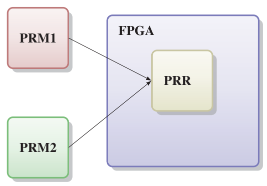

Fig. 1. Partial Reconfiguration in FPGAs

In figure 1, we see that two PRMs which are mutually exclusive in time will be placed in the PRR inside the FPGA i.e., only one PRM can be assigned to a given PRR at a given time. The remaining region in the FPGA which is outside the PRR is the static region, where the application which needs to be run uninterruptedly, is placed. The configuration files placed in the PRR are called as partial bitfiles. In FPGAs supporting dynamic PR, single configuration units can be read, modified and written. For example in Xilinx FPGAs, different Virtex-Series support different PR schemes. Virtex-II/Virtex-II Pro supported initially column-based and later also tile-based PR. The high-end FPGAs like Virtex-4 and Virtex-5 support tile-based PR only. The partial reconfiguration of an FPGA is done through the internal configuration access port (ICAP), a built-in hard core IP module available on the FPGA. ICAP gives internal access to Select MAP configuration interface and can process every bitfile that could also be used externally. The ICAP module is controlled by the software driver for the processor on the FPGA. In our scenario, the static logic contains the asymmetric algorithm (RSA or ECC), which generates the public-private key pair. The reconfigurable logic is populated with the partial bitstreams, which are the actual FPGA applications to be implemented, after an on-board decryption process.

3.2. Public Key Cryptography Overview

Public key cryptography uses asymmetric key algorithms like RSA and ECC. Unlike symmetric key algorithms, they do not require a secure initial exchange of key between the sender and the receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key, which is known only to the receiver. Using the above idea, in our scenario the host PC (sender), which has the stored bitfiles to be loaded on to the FPGA (receiver), encrypts them with the public key received from the FPGA and sends them back to the FPGA, which decrypts the bitfiles with the private key located on-board, and triggers the configuration.

4. PR AND PKC BASED FPGA BITSTREAM PROTECTION

4.1. Assumptions for the Proposed Model

The main objective of the proposed scheme is to protect the FPGA configuration files without the fixed storage of keys in/out of FPGA. We prefer SRAM-based FPGAs over Flash-based as the latter are not in common use yet. The assumed scenario for the proposed scheme is inhouse, i.e., the PC and the FPGA are directly connected to each other. It is assumed that there is no adversary, who is trying to intercept the communication channel between the host PC and the FPGA. But if the loading of the bitstream from a remote area over the Internet is to be considered, then there is a possibility of several attacks such as the well-known “man-in-the-middle” attack. Here, the attacker can pose himslef to be the FPGA and send his own public key to the host PC. Thereby, decrypting the incoming bitstream with his generated private key, sufficient to unveil the application to be run on the FPGA.

4.2. Methodology

A very secure method for protecting the FPGA configuration files can be built when PR and PKC are combined. By using these two basic methods we propose a novel technique to protect FPGA IP without the need to store any cryptographic keys in a dedicated storage. In figure 2, we see that all of the functions in the "blue colored box" can be implemented with in the physical package of the FPGA. The plaintext and the private key information never leave a well-protected container i.e., the security boundary.

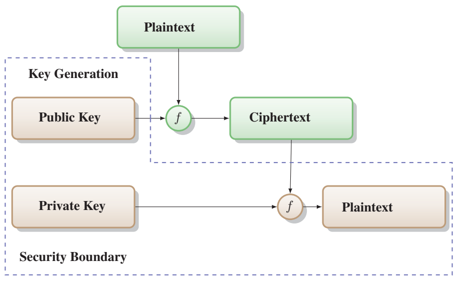

Fig. 2. Asymmetric Key Encryption

The architecture of the proposed method is outlined in figure 4 and the corresponding communication protocol is given in figure 3. The considered FPGA supports the dynamic PR, and we divide the FPGA's logic into two parts: static area and dynamic area. The initial bitfile (full bitstream) to be loaded onto the FPGA in the static area is an unencrypted design that does not feature any proprietary information. It only contains the algorithm to generate the public-private key pair and the interface between the host, FPGA, and ICAP. The configuration of the FPGA is done according to the protocol illustrated in figure 3.

Following advantages result from using the proposed scheme:

- The public-private key pair may be regenerated at any time. If a new configuration is downloaded from the host it may be encrypted with a different public key and decrypted with the corresponding new private key. Even if the FPGA is configured with the same partial bitfile later, such as after a power-on-reset, a different public key pair is used even though it is the same bitfile.

- There is no need of any non-volatile memory to store the key (as for symmetric keys in previously mentioned schemes) for decrypting the bitfiles as the private key is generated on the fly. In addition, the private key generated by the asymmetric algorithm running on the FPGA is stored in the SRAM and if the FPGA loses power then the private key no longer exists.

- In general, the partial bitfile contains the vast majority of the FPGA design with the logic in the static design consuming a very small percentage of the overall FPGA resources. So, most of the FPGA resources are allocated to the applications.

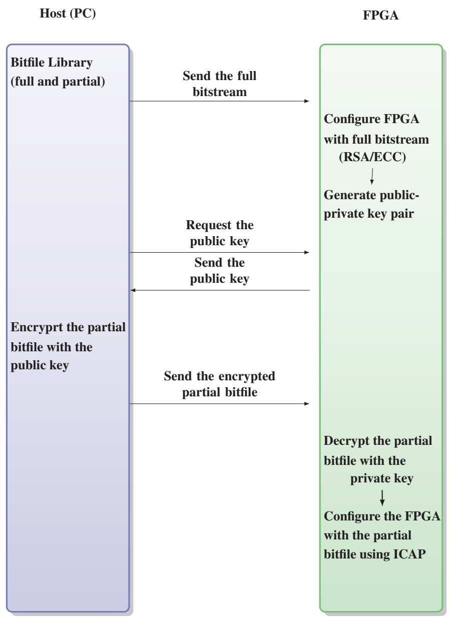
Host (PC)
FPGA
Bitfile Library (full and partial)
Send the full bitstream
Configure FPGA with full bitstream (RSA/ECC)
Generate public-private key pair
Request the public key
Send the public key
Encryprt the partial bitfile with the public key
Send the encrypted partial bitfile
Decrypt the partial bitfile with the private key
Configure the FPGA with the partial bitfile using ICAP

Fig. 3. Protocol for secure FPGA Configuration

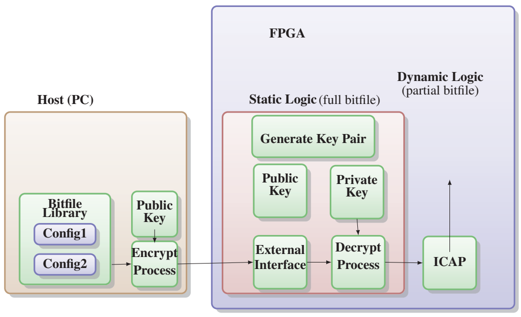

Fig. 4. Loading an encrypted partial Bitfile

- Even if at some point of time it is found that the asymmetric algorithm being used is no more secure, one can replace it with a new algorithm as it just requires loading a new full bitstream into the static region of the FPGA.

- Even if the system is stolen and the FPGA remains powered it is extremely difficult to find the private key, because it is stored in the general purpose FPGA fabric, but not in a special purpose register.

- The issue of loading IPs from different vendors onto a single SoC is addressed.

- This scheme can be applied to low-end FPGAs too, which support the partial reconfiguration.

There are also certain disadvantages with this scheme, like the implementation of asymmetric key algorithm

Table 1. ECC, AES and RSA Implementations

Algorithm	LUTs	Registers	BRAMs	Delay
ECC	2466	1207	2	5.142 ns
AES	853	536	5	3.870 ns
RSA	16319	12080	2	7.442 ns

(RSA) on an FPGA consumes a lot of resources and the scenario is local with a set of assumptions. Although the generation of partial and full bitstreams for the FPGA is cost expensive, time expensive, and exhausting at the moment, the FPGA vendors claim that it will be much easier on top of their newer versions of tools.

5. IMPLEMENTATION RESULTS

Algorithms ECC, AES, and RSA have been implemented on a Xilinx V5LX110T platform and their resource requirements are compared to show the feasibility of implementation of the proposed scheme. The resource consumption for the algorithms are summarised in table 1. Obviously, the number of resources (slice LUTs and slice registers) occupied by the public key algorithm (RSA) is much higher compared to the symmetric key algorithm (AES), but the ECC resource consumption is comparable with AES. Also, the calculation time delay for each of the algorithms is measured at a speed grade of -1 of the FPGA devices. So, The use of the ECC algorithm for decrypting the incoming encrypted partial bitstream instead of on-board AES decryption unit is justified with reference to the overall advantages gained as mentioned in the previous section.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a novel design method to protect the FPGA configuration files which avoids the need to store the cryptographic keys in registers of the FPGA or in an external non-volatile memory. The proposed scheme uses the special feature of SRAM-based FPGAs, i.e., dynamic partial reconfiguration and the well-known public key cryptography scheme to secure the IP of the design. This scheme can be further extended to secure the IPs supplied from different vendors. The feasibility of implementation of the proposed scheme on a Xilinx Virtex-5 FPGA platform and some of the implementation results were presented. As a part of future work there is a need to reduce the number of resources being consumed by public key algorithms through algorithm optimization. In addition, we will consider how to avoid the man-in-the-middle attack, which is not addressed in this paper.

7. REFERENCES

[1] S. Mangard, "A simple power-analysis (spa) attack on implementations of the aes key expansion," in *ICISC'02: Proceedings of the 5th international conference on Information security and cryptology*. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 343–358.

[2] R. Anderson and M. Kuhn, "Tamper resistance: a cautionary note," in *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, 1996, pp. 1–1.

[3] ——, "Low cost attacks on tamper resistant devices." Springer-Verlag, 1997, pp. 125–136.

[4] in *Xilinx Corporation. Virtex-II platform FPGA Handbook.*

[5] in *Xilinx Corporation. Virtex-5 FPGA configuration guide.*

[6] T. Kean, "Secure configuration of field programmable gate arrays," in *2001 International Conference on Field Programmable Logic and Applications*. Springer-Verlag, 2001, pp. 142–151.

[7] K. Yip and T. Ng, "Partial-encryption technique for intellectual property protection of FPGA-based products," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 183–190, Feb. 2000.

[8] J. Guajardo, S. S. Kumar, G. Schrijen, and P. Tuyls, "Physical unclonable functions and Public-Key crypto for FPGA IP protection," in *2007 International Conference on Field Programmable Logic and Applications*, 2007, pp. 189–195.

[9] J. Lach, W. H. Mangione-smith, and M. Potkonjak, "Signature hiding techniques for fpga intellectual property protection," in *1998 IEEE/ACM International Conference on Computer Aided design*, 1998.

[10] T. Guneysu, B. Moller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *2007 International Conference on Field Programmable Technology*, 2007, pp. 169–176.

[11] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, Santa Fe, NM, USA, pp. 146–153.